



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1800  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
-----------------	-------------	----------------------	---------------------	------------------

10/722,761

11/26/2003

Charles S. Shorb

343355600055

7202

7590

06/19/2006

John V. Biernacki  
Jones Day  
North Point  
901 Lakeside Avenue  
Cleveland, OH 44114

EXAMINER

TSAI, SHENG JEN

ART UNIT

PAPER NUMBER

2186

DATE MAILED: 06/19/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

# Office Action Summary

Application No.

10/722,761

Applicant(s)

SHORB, CHARLES S.

Examiner

Sheng-Jen Tsai

Art Unit

2186

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

## Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

## Status

- 1) ☒ Responsive to communication(s) filed on 26 November 2003.
- 2a) ☐ This action is FINAL. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

## Disposition of Claims

- 4) ☒ Claim(s) 1-42 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-42 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

## Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

## Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
  - ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

## Attachment(s)

- ☒ Notice of References Cited (PTO-892)
- ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)  
Paper No(s)/Mail Date 11/26/03, 03/09/04
- ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date. \_\_\_\_\_
- ☐ Notice of Informal Patent Application (PTO-152)
- ☐ Other: \_\_\_\_\_

### DETAILED ACTION

1. Claims 1-42 are presented for examination in this application (10,722,761) filed on November 26, 2003.

Acknowledge is made of information disclosure document filed on 3/19/2004 and 11/26/2003.

### *Claim Rejections - 35 USC § 102*

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

2. Claims 1, 3-24 and 40-42 are rejected under 35 U.S.C. 102(b) as being anticipated by Oliver (U.S. 6,029,190).

As to claim 1, Oliver discloses a **computer-implemented shared locking data store** [Read Lock and Write Lock management system Based Upon MUTEX and Semaphore Availability (title)] **for handling multiple executable entities' access to at least one resource** [in an operating environment that allows objects such as threads to wait on a plurality of synchronization objects simultaneously (abstract); One of the challenges in building a multi-threaded application or a multi-tasking application is controlling access to per-process protected data, particularly file handles or dynamically-allocated data structures (column 1, lines 13-15)], **said shared locking data store being stored in a computer memory** [As described above, the traditional model of a single mutex handling read/write access for multiple threads is deficient. Thus, the present invention introduces a read/write lock mechanism by making use additionally of a second synchronization object known as a semaphore as well as a tracking variable. The combined use of semaphore, mutex and tracking variable allows

Art Unit: 2186

efficient, stable multi-thread data access in a more simultaneous fashion than the prior art (column 2, lines 52-59)], **said shared locking data store comprising:**

**write requested data that is indicative of when a write request for the resource has been made** [figure 2, write lock, step 220, "wait until MUTEX and Semaphore are simultaneously available;" which means that the write request must wait; column 4, lines 20-30 show a plurality of threads, including a write thread, are waiting on the resource];

**writer active data that is indicative of whether a writer process is actively undergoing a write operation with respect to the resource** [As described above, the traditional model of a single mutex handling read/write access for multiple threads is deficient. Thus, the present invention introduces a read/write lock mechanism by making use additionally of a second synchronization object known as a semaphore as well as a tracking variable. The combined use of semaphore, mutex and tracking variable allows efficient, stable multi-thread data access in a more simultaneous fashion than the prior art (column 2, lines 52-59); figures 1 and 2 show how the combined values of MUTEX, Semaphore and the tracking variable (NUMBERREADERS) provide the indication of an active write operation];

**wait event posted data, wherein the wait event posted data is indicative of whether a read request to the resource has completed, wherein the wait event posted data may be used to indicate when a write request can access the resource** [As described above, the traditional model of a single mutex handling read/write access for multiple threads is deficient. Thus, the present invention

introduces a read/write lock mechanism by making use additionally of a second synchronization object known as a semaphore as well as a tracking variable. The combined use of semaphore, mutex and tracking variable allows efficient, stable multi-thread data access in a more simultaneous fashion than the prior art (column 2, lines 52-59); particularly, figures 1 and 2 show how the combined values of MUTEX, Semaphore and the tracking variable (NUMBERREADERS) provide the indication of whether a read request has completed (NUMBERREADERS = 0) and the value of MUTEX and Semaphore are used to indicate when a write request can access the resource (step 210 of figure 2)];

**reader data that is indicative of whether a reader process is active and attempting to read from the resource** [the combined value of MUTEX, Semaphore and the tracking variable (NUMBERREADERS) indicate whether a reader process is active and attempting to read from the resource (step 118 and 134 of figure 1); column 2, lines 60-67; columns 3-4];

**wherein the shared locking data store allows writer and reader locking state information to be determined so that access to the resource may be handled** [the combined values of MUTEX, Semaphore and the tracking variable (NUMBERREADERS) provide the write and read lock state information as shown in figures 1 and 2; column 2, lines 60-67; columns 3-4].

As to claim 3, Oliver teaches that **the multiple executable entities include threads which are to access the resource** [in an operating environment that allows objects such as threads to wait on a plurality of synchronization objects simultaneously

(abstract); One of the challenges in building a multi-threaded application or a multi-tasking application is controlling access to per-process protected data, particularly file handles or dynamically-allocated data structures (column 1, lines 13-15)], **wherein the threads comprise a writer thread and a reader thread** [figures 1 and 2; column 4, lines 20-30 show a plurality of threads, including a write thread, are waiting on the resource], **wherein the locking state information is used to determine how the writer thread and the reader thread access the resource** [the combined values of MUTEX, Semaphore and the tracking variable (NUMBERREADERS) provide the write and read lock state information as shown in figures 1 and 2; column 2, lines 60-67; columns 3-4].

As to claim 4, Oliver teaches that **the resource requires protection on a per thread basis** [One of the challenges in building a multi-threaded application or a multi-tasking application is controlling access to per-process protected data, particularly file handles or dynamically-allocated data structures (column 1, lines 13-15)].

As to claim 5, Oliver teaches that **the locking state information indicates where in the overall locking process an operating system is with respect to the resource** [the combined values of MUTEX, Semaphore and the tracking variable (NUMBERREADERS) provide the write and read lock state information as shown in figures 1 and 2; column 2, lines 60-67; columns 3-4].

As to claim 6, Oliver teaches that **the request of the reader thread is allowed to succeed unless there is a write request active or pending as indicated by the write requested data and the writer active data** [the combined values of MUTEX,

Semaphore and the tracking variable (NUMBERREADERS) provide the write and read lock state information as shown in figures 1 and 2; figure 1, step 110 show that a read request is allowed to proceed if MUTEX is available, otherwise it must wait for MUTEX to be released by an on-going write process (step 260, figure 2); column 2, lines 60-67; columns 3-4].

As to claim 7, Oliver teaches that **rules mechanism that indicates how resource access requests are to be handled based upon the data stored in the shared locking data store** [the combined values of MUTEX, Semaphore and the tracking variable (NUMBERREADERS) provide the write and read lock state information as shown in figures 1 and 2; column 2, lines 60-67; columns 3-4].

As to claim 8, Oliver teaches that **the rules mechanism includes allowing any number of read requests to succeed unless there is a writer active or pending as indicated by the shared locking data store** [Multiple reader threads may consecutively obtain the mutex (step 114) and read out data, until a write supersedes and claims the mutex (see FIG. 2) (column 3, lines 18-20); The mutex is only available when all priority protected writes on that location have been performed (see FIG. 2). If the mutex is not available, then a wait state is employed (step 112) until the mutex does become available (column 2, lines 65-67)].

As to claim 9, Oliver teaches that **the rules mechanism includes allowing only one writer to be active at any given time** [When there are multiple writer threads, these writer threads are not permitted to own the write lock simultaneously (column 3,

lines 61-63); figure 2, step 210 ensures that only one write thread is allow to perform write operation at a time].

As to claim 10, Oliver teaches that **the rules mechanism includes that no preference is given when deciding which of waiting read or write requests should be honored first** [the procedures shown in figures 1 and 2 do not give preference to either read or write requests]

As to claim 11, Oliver teaches that **the rules mechanism substantially eliminates a reader request starvation or a writer request starvation situation with respect to the resource** [the procedures shown in figures 1 and 2 do not give preference to either read or write requests, is fair to both read and write requests, hence prevent starvation of either read requests or write requests].

As to claim 12, Oliver teaches **the resource comprises data stored in computer memory** [One of the challenges in building a multi-threaded application or a multi-tasking application is controlling access to per-process protected data, particularly file handles or dynamically-allocated data structures (column 1, lines 13-15)].

As to claim 13, Oliver teaches that **the resource comprises a computer file** [One of the challenges in building a multi-threaded application or a multi-tasking application is controlling access to per-process protected data, particularly file handles or dynamically-allocated data structures (column 1, lines 13-15)].

As to claim 14, Oliver teaches that **the resource comprises an input/output device** [One of the challenges in building a multi-threaded application or a multi-tasking



application is controlling access to per-process protected data, particularly file handles or dynamically-allocated data structures (such as data from an input/output device) (column 1, lines 13-15)].

As to claim 15, Oliver teaches that **the write requested data is to have a set status when a write request has been made** [figure 2, using the MUTEX].

As to claim 16, Oliver teaches that **a set status for the write requested data indicates whether an operating system (OS) mutex lock is to be used for processing access to the resource** [figure 2; The combined use of semaphore, mutex and tracking variable allows efficient, stable multi-thread data access in a more simultaneous fashion than the prior art (column 2, lines 52-59)].

As to claim 17, Oliver teaches that **indication of whether a writer process is active by the writer active data is used to protect against readers posting an operating system (OS) event after the writer thread has been activated** [figure 1, step 110 show that a read request is allowed to proceed if MUTEX is available, otherwise it must wait for MUTEX to be released by an on-going write process (step 260, figure 2); column 2, lines 60-67; columns 3-4].

As to claim 18, Oliver teaches that **the shared locking data store of claim 1 further comprising wait event posted data, wherein the wait event posted data is indicative of whether a read request to the resource has completed, wherein the wait event posted data may be used to indicate when a write request can access the resource, wherein a last active read clears status of the wait event posted data prior to posting an event** [figures 1 and 2 show how the combined values of

MUTEX, Semaphore and the tracking variable (NUMBERREADERS) provide the indication of whether a read request has completed ( $\text{NUMBERREADERS} = 0$ ) and the value of MUTEX and Semaphore are used to indicate when a write request can access the resource (step 210 of figure 2); step 138 of figure 1 shows that the last read event clear the MUTEX].

As to claim 19, Oliver teaches that **the clearing ensures that one and only one posting to a writer of an event occurs** [When there are multiple writer threads, these writer threads are not permitted to own the write lock simultaneously (column 3, lines 61-63); figure 2, step 210 ensures that only one write thread is allow to perform write operation at a time].

As to claim 20, Oliver teaches that **the posting occurs if status of the writer active data is not set** [figure 2, step 210, both MUTEX and Semaphore must be available].

As to claim 21, Oliver teaches that **the processes comprise threads, wherein the wait event posted data indicates when events may be posted by one thread to notify another thread that the other thread's request may be processed** [figure 1, steps 116 and 132; the value of the tracking variable NUMBERREADERS gives indication that other read threads request are still in progress].

As to claim 22, Oliver teaches that **the wait event posted data indicates whether a reader thread can post a lock release event to a writer thread** [figure 1, steps 116 and 132; the value of the tracking variable NUMBERREADERS gives indication that other read threads request are still in progress, if NUMBERREADERS is

not zero, the MUTEX will not be released (step 138), which would prevent a write event from occurring (figure 2, step 210)].

As to claim 23, Oliver teaches that **the wait event posted data is used to prevent multiple reader threads from posting redundant lock release events** [The combined use of semaphore, mutex and tracking variable allows efficient, stable multi-thread data access in a more simultaneous fashion than the prior art (column 2, lines 52-59); figure 1, step 122 shows that multiple read threads are allow to proceed using only one MUTEX].

As to claim 24, Oliver teaches that **the write requested data, writer active data, and wait event posted data provide multi-threaded protection in place of an operating system (OS) mutex** [As described above, the traditional model of a single mutex handling read/write access for multiple threads is deficient. Thus, the present invention introduces a read/write lock mechanism by making use additionally of a second synchronization object known as a semaphore as well as a tracking variable. The combined use of semaphore, mutex and tracking variable allows efficient, stable multi-thread data access in a more simultaneous fashion than the prior art (column 2, lines 52-59)].

As to claim 40, Oliver discloses **a computer-implemented method** [figures 1 and 2 show the flowcharts of computer programs implementing the lock management algorithms] **for handling multiple executable entities' access to at least one resource** [in an operating environment that allows objects such as threads to wait on a plurality of synchronization objects simultaneously (abstract); One of the challenges in

building a multi-threaded application or a multi-tasking application is controlling access to per-process protected data, particularly file handles or dynamically-allocated data structures (column 1, lines 13-15)], **comprising:**

**receiving a request from an executable entity to access a resource** [figures 1 and 2 show read and write requests from threads, respectively];

**determining how to process the resource request based upon lock state data** [the combined values of MUTEX, Semaphore and the tracking variable (NUMBERREADERS) provide the write and read lock state information as shown in figures 1 and 2; column 2, lines 60-67; columns 3-4];

**wherein lock state data comprises write requested data and reader data** [the combined values of MUTEX, Semaphore and the tracking variable (NUMBERREADERS) provide the write and read lock state information as shown in figures 1 and 2; column 2, lines 60-67; columns 3-4];

**wherein the write requested data is indicative of when a write request for the resource has been made with respect to the resource** [figure 2, write lock, step 220, "wait until MUTEX and Semaphore are simultaneously available;" which means that the write request must wait; column 4, lines 20-30 show a plurality of threads, including a write thread, are waiting on the resource];

**wherein if the write requested data indicates that a write request has not been made for the resource, then providing access to the resource such that an operating system (OS) mutex is avoided for a read lock acquisition** [figure 1, steps 110 and 114] ;

**wherein the reader data is indicative of whether a reader executable entity is active and attempting to read from the resource** [As described above, the traditional model of a single mutex handling read/write access for multiple threads is deficient. Thus, the present invention introduces a read/write lock mechanism by making use additionally of a second synchronization object known as a semaphore as well as a tracking variable. The combined use of semaphore, mutex and tracking variable allows efficient, stable multi-thread data access in a more simultaneous fashion than the prior art (column 2, lines 52-59); figures 1 and 2 show how the combined values of MUTEX, Semaphore and the tracking variable (NUMBERREADERS) provide the indication of an active write operation].

As to claim 41, refer to "As to claim 40" presented earlier in this Office Action.

As to claim 42, Oliver teaches that **the apparatus of claim 41 further comprising:**

**read lock acquisition means for accessing the lock state data in order to process a read lock acquisition of the resource** [figure 1 shows how to acquire and release a read lock];

**read lock release means for accessing the lock state data in order to process a read lock release of the resource** [figure 1 shows how to acquire and release a read lock];

**write lock acquisition means for accessing the lock state data in order to process a write lock acquisition of the resource** [figure 2 shows how to acquire and release a write lock];

**write lock release means for accessing the lock state data in order to process a write lock release of the resource** [figure 2 shows how to acquire and release a write lock].

***Claim Rejections - 35 USC § 103***

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

4. Claim 2 is rejected under 35 U.S.C. 103(a) as being unpatentable over Oliver (U.S. 6,029,190).

As to claim 2, Oliver teaches that **the executable entities include processes, threads, stand-alone application or code that runs at the request of another program** [One of the challenges in building a multi-threaded application or a multi-tasking application is controlling access to per-process protected data, particularly file handles or dynamically-allocated data structures (such as data from an input/output device) (column 1, lines 13-15)] **wherein use of the shared locking data stores allows for the avoidance of an operating system call or avoidance of a resource accessing trap of the operating system's kernel** [figures 1 and 2 show that the management of read lock and the write lock does not rely on either operating system call, nor does it invoke any trap of the operating system kernel; column 2, lines 60-67; columns 3-4].

Regarding claim 2, Oliver does not mention that **the executable entities include daemons, applets, servlets or ActiveX components.**

However, these entities are well known in the art and have been widely used in computer systems (see Microsoft Computer Dictionary, 5<sup>th</sup> edition, Microsoft Press, 2002, isbn 0-7356-1495-4; page 140 – daemon; page 31 – applet; page 18 – ActiveX; page 475 – servlet).

Therefore, it would have been obvious for one of ordinary skills in the art at the time of Applicant's invention to recognize that these entities are well known in the art and have been widely used in computer systems, hence lacking patentable significance.

5. Claims 25-38 are rejected under 35 U.S.C. 103(a) as being unpatentable over Oliver (U.S. 6,029,190), and in view of Ofer (US 6,718,448).

As to claim 25, Oliver teaches that **the locking state data comprises the write requested data, the writer active data, the reader count data, and the wait event posted data** [As described above, the traditional model of a single mutex handling read/write access for multiple threads is deficient. Thus, the present invention introduces a read/write lock mechanism by making use additionally of a second synchronization object known as a semaphore as well as a tracking variable. The combined use of semaphore, mutex and tracking variable allows efficient, stable multi-thread data access in a more simultaneous fashion than the prior art (column 2, lines 52-59); the combined values of MUTEX, Semaphore and the tracking variable

(NUMBERREADERS) provide the write and read lock state information as shown in figures 1 and 2; column 2, lines 60-67; columns 3-4].

Regarding claim 25, Oliver does not explicitly refer the combination of MUTEX, Semaphore and variable NUMBERREADERS illustrated in figures 1 and 2 as a single atomic unit.

However, the process and procedure disclosed in figures 1 and 2 on managing the read lock and write lock [column 2, lines 60-67; columns 3-4] are essentially an atomic operation because the use of MUTEX and Semaphore would prevent other threads from interrupting the activities of the thread currently possesses MUTEX and Semaphore.

Further, Ofer discloses in the invention "Queued Locking of a Shared Resource Using Multimodal Lock Types" a lock mechanism including a main lock data structure which provides, in a single atomic structure, the resources needed to lock the shared resource [column 3, lines 17-19].

Atomic structure and operations are critical in maintaining consistency when locks are used in an environment where resources are shared among a plurality of entities [column 1, lines 54-67; column 2, lines 1-4].

Therefore, it would have been obvious for one of ordinary skills in the art at the time of Applicant's invention to recognize the crucial nature of atomic structure and operation in an environment where resources are shared among a plurality of entities, as demonstrated by Ofer, and to make the explicit connections between the teaching



from Ofer and what is illustrated by Oliver in figures 1 and 2 as essentially atomic operations.

As to claim 26, refer to "As to claim 25" for the explanation of "atomic" structure and operations. Further, Oliver's shared locking data comprises MUTEX, Semaphore and variable NUMBERREADERS, which constitutes a **multi-bit unit**.

As to claim 27, refer to "As to claim 25" for the explanation of "atomic" structure and operations.

As to claims 28-31, refer to "As to claim 25" for the explanation of "atomic" structure and operations. Further, Oliver's shared locking data comprises MUTEX, Semaphore and variable NUMBERREADERS that would have to be updated for every thread read/write request [figures 1 and 2]. Updating MUTEX, Semaphore and variable NUMBERREADERS necessitates reading (i.e., get operation) their values, modifying (i.e., incrementing NUMBERREADERS, an add operation; setting the value to a "1" or clearing the value to a "0") their values and writing (i.e., sub) the modified values. Since all these operations are performed in an atomic manner, as explained in "As to claim 25," they are essentially **atomic get, atomic set, atomic add and atomic sub operation**, respectively.

As to claim 32, Oliver teaches that **the locking state data is formatted as a multi-bit single unit** [refer to "As to claim 26"];  
**wherein the write requested data comprises a single bit in the unit** [figure 2, step 220 signals a "waiting write request"];

**wherein the writer active data comprises a single bit in the unit** [the corresponding single bit is the MUTEX flag (figure 2, step 230)];

**wherein the wait event posted data comprises a single bit in the unit** [the corresponding single bit is the Semaphore flag (figure 1, steps 120 and 136; figure 2, steps 230 and 240)];

**wherein the reader data comprises a plurality of bits in the unit to indicate number of active readers with respect to the resource** [the corresponding a plurality of bits entity is the variable NUMBERREADERS (figure 1, steps 116 and 132)].

As to claim 33, refer to "As to claim 32."

As to claim 34, Oliver teaches that **the reader data's count of the readers provides an indication as to whether readers are present and when the last reader exits** [figure 1; the tracking variable (NUMBERREADERS) provide the indication of whether a read request has completed (NUMBERREADERS = 0); the tracking variable (NUMBERREADERS) indicate whether a reader process is active and attempting to read from the resource (step 118 and 134 of figure 1); column 2, lines 60-67; columns 3-4].

As to claim 35, Oliver teaches that **a read lock acquisition means accesses the locking state data for processing a read lock acquisition of the resource** [figure 1 shows how to acquire and release a read lock].

As to claim 36, Oliver teaches that **a read lock release means accesses the locking state data for processing a read lock release of the resource** [figure 1 shows how to acquire and release a read lock].

As to claim 37, Oliver teaches that **a write lock acquisition means accesses the locking state data for processing a write lock acquisition of the resource** [figure 2 shows how to acquire and release a write lock].

As to claim 38, Oliver teaches that **a write lock release means accesses the locking state data for processing a write lock release of the resource** [figure 2 shows how to acquire and release a write lock].

As to claim 39, Oliver teaches that **the operating system returns a lock busy status indicator when a lock cannot be obtained within a predetermined time** [the MUTEX and Semaphore serve as the busy flag to indicate that the lock can not be obtained (figure 1, steps 110 and 126; figure 2 step 210)].

## **6. *Related Prior Art of Record***

The following list of prior art is considered to be pertinent to applicant's invention, but not relied upon for claim analysis conducted above.

- McClaughry et al., (US 5,933,825), "Arbitrating Concurrent Access to File system Objects."
- McKenney, (US Patent Application Publication 2004/0117531), "Adaptive Reader-Writer Lock."
- Kakivaya et al., (US 6,546,443), "Concurrency-Safe Reader-Writer Lock with Time Out Support."

- Clark, (US 6,598,068), "Method and Apparatus for Automatically Managing Concurrent Access to a Shared Resource in a Multi-Threaded Programming Environment."
- Bacon, (US 6,247,025), "Locking and Unlocking Mechanism for Controlling Concurrent Access to Objects."
- Watanabe et al., (US 6,016,490), "Database Management System."
- Harris, (US Patent Application Publication 2003/0200398), "Method and Apparatus for Emulating Shared Memory in a Storage Controller."
- Onodera et al., (US 6,883,026), "Method and Apparatus for Managing Locks of Objects and Method and Apparatus for Unlocking Objects."
- Won et al., (US Patent Application Publication 2003/0126187), "Method and Apparatus for Synchronization in a Multi-Thread system of JAVA Virtual machine."
- Li, (US Patent Application Publication 2004/0059733), "Method and Apparatus for Locking Objects in a Multi-Threaded Environment."
- Li, (US Patent Application Publication 2003/0233393), "Thread Optimization for Lock and Unlock operations in a Multi-Threaded Environment."

### ***Conclusion***


7. Claims 1-42 are rejected as explained above.
8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Sheng-Jen Tsai whose telephone number is 571-272-4244. The examiner can normally be reached on 8:30 - 5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Matthew Kim can be reached on 571-272-4182. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Sheng-Jen Tsai  
Examiner  
Art Unit 2186

April 26, 2006

  
MATTHEW KIM  
SUPERVISORY PATENT EXAMINER  
TECHNOLOGY CENTER 2186